SwarMer: A Decentralized Localization Framework for Flying Light Specks

Hamed Alimohammadzadeh, Shahram Ghandeharizadeh
Computer Science Department, University of Southern California
Los Angeles, California, USA
{halimoha,shahram}@usc.edu

ABSTRACT

Swarm-Merging, SwarMer, is a decentralized framework to localize Flying Light Specks (FLSs) to render 2D and 3D shapes. An FLS is a miniature sized drone equipped with one or more light sources to generate different colors and textures with adjustable brightness. It is battery powered, network enabled with storage and processing capability to implement a decentralized algorithm such as SwarMer. An FLS is unable to render a shape by itself. SwarMer uses the inter-FLS relationship effect of its organizational framework to compensate for the simplicity of each individual FLS, enabling a swarm of cooperating FLSs to render complex shapes. SwarMer is resilient to network packet loss, FLSs failing, and FLSs leaving to charge their battery. It is fast, highly accurate, and scales to remain effective when a shape consists of a large number of FLSs.

Holodecks Reference Format:
Hamed Alimohammadzadeh, Shahram Ghandeharizadeh. SwarMer: A Decentralized Localization Framework for Flying Light Specks . Holodecks, 1(1): 10-22, 2023.
doi:10.61981/ZFSH2302

Holodecks Artifact Availability:
SwarMer source code is available at https://github.com/flyinglightspeck/SwarMer. See https://youtu.be/BIiBxD_aUz8 for a MATLAB demonstration of SwarMer, https://youtu.be/Lh11tWWOP5Y for two relative localization techniques as SwarMer plugins. SwarMer is able to transition FLSs from illuminating one point cloud to the next point cloud, see https://youtu.be/4GhhlSq4UrM. It may start with a random pattern and illuminate a shape, see https://youtu.be/cZrz0e61txU. The reported implementation of SwarMer localizes FLSs into a Hat (https://youtu.be/y_xHdz5bs5M), a Dragon (https://youtu.be/PRyjdmw8NVc), and a Skateboard (https://youtu.be/a0fFu0z6BU0).

1 INTRODUCTION

An FLS is a miniature sized drone equipped with one or more light sources to generate different colors and textures with adjustable brightness. Synchronized swarms of FLSs will illuminate virtual objects in a pre-specified 3D volume, an FLS display [30, 31]. An FLS display may be a cuboid that sits on a table or hangs on a wall, the dashboard of a self-driving vehicle, a room, etc. It will render static 2D or 3D point clouds by assigning the coordinates of their points to different FLSs to illuminate.

Localizing FLSs to render a 2D or 3D point cloud is non-trivial for several reasons. First, in an indoor setting, FLSs may lack line of sight with GPS satellites to position themselves. Even if GPS was available, it would be too coarse [30, 47, 71]. The best some GPS devices claim is a 3 meter accuracy 95% of the time [47, 77].

Second, a dead-reckoning technique that requires an FLS to fly to its assigned position is known to suffer from errors that increase as a function of the distance traveled by a drone [13, 14, 18, 73]. Greater distances result in higher errors, distorting the final rendering. To illustrate, Figure 1.b shows rendering of three 2D point clouds of Figure 1.a (ground truth) using dead reckoning with 5 degrees of error. (See Section 4.1 for a formal definition of the *degree of error*.) The FLSs are deployed by a dispatcher at the origin, (0, 0). Each FLS uses a collision free path [5, 6, 8, 12, 15, 19, 20, 26, 27, 36, 37, 39, 44, 52, 53, 55, 62, 63, 76, 78, 81–84, 99] to travel to its coordinate. With dead reckoning, the wing of the butterfly farthest from (closest to) the origin is least (most) accurate because FLSs travel a greater (shorter) distance. With 5 degrees of error, the butterfly wing farthest from the origin is distorted completely, see Figure 1.b. In general, dead reckoning is challenged with realizing symmetrical geometric shapes as simple as a straight line, e.g., the cat's tail which is close to the dispatcher.

The challenge is to design and implement a general purpose framework that localizes FLSs to render any 2D or 3D shape with an arbitrary amount of dead-reckoning error. This challenge constitutes the focus of this paper. The ideal framework must satisfy the following requirements. First, it must be fast by allowing FLSs to localize as they are deployed. This results in an incremental rendering where portions of a shape become recognizable as other FLSs are deployed to render the remainder of the shape. Second, it should be highly accurate. A rendering should match its point cloud with 100% precision. With some applications, this may be overkill because the human eye may not detect one FLS out of its intended position even with a small point cloud requiring one hundred FLSs, e.g., compare Figure 1.c with 1.a. Third, it should be continuous as drones are known to drift [28, 49]. FLSs must be able to execute the algorithm to compensate for both their drift and those of their neighbors. Fourth, it should be resilient to FLS failures and departures. FLSs are battery powered mechanical devices that may fail [30]. An FLS may fly back to a charging station per an algorithm such as STAG [31]. The substituting FLS should localize itself. Fifth, it should scale to render shapes consisting of a large number of FLSs, tens of thousands if not millions. Finally, it should be minimal, requiring as few sensors as possible from the display to enhance its flexibility and reduce its cost.

Proceedings of the Holodecks Foundation, Vol. 1, No. 1.
doi:10.61981/ZFSH2302

Swarm-Merging, SwarMer, addresses these requirements for 2D and 3D point clouds. It is designed for an in-door 3D display that lacks line of sight with GPS satellites. It is a *relative* localization system instead of an *absolute* one, e.g., the teapot of Figure 1.c is moved down relative to its ground truth shown in Figure 1.a. Given a 2D or 3D point cloud, it approximates the position of the FLSs by forcing the neighboring FLSs to match the relative distance and angle [14] of their assigned points. Even though the location of each FLS is not exactly the same as that specified by the point cloud, SwarMer's relative positioning of FLSs produces shapes true to a point cloud. In some instances, the relative positioning matches the coordinates of the points in a point cloud exactly.

SwarMer is a decentralized algorithm, requiring each FLS to localize itself relative to another FLS at a time. Many FLSs may correct their position concurrently and independently. Hence, SwarMer may render a portion of a shape while FLSs are being deployed to complete the rest of the shape. This makes SwarMer fast.

SwarMer constructs swarms [66] of FLSs. Movement of one FLS causes all FLSs in its swarm to move. SwarMer includes provisions that prevent local minimums that result in isolated FLSs or fragmented swarms when there should be one swarm. Hence, SwarMer is resilient to FLSs failing and departing to charge their battery.

SwarMer requires FLSs to position themselves relative to one another to render the intended shape with a high accuracy. Figure 1.c shows localization of FLSs with SwarMer. This positioning uses dead-reckoning with the same 5 degrees of error. The distance between FLSs is typically shorter than the distance from the dispatcher located at the origin. Hence, error with inter-FLS dead-reckoning decreases, enhancing SwarMer's accurracy.

Figures 1.d and 1.e show the use of triangulation and trilateration to localize FLSs. Both are designed for use with anchors that have precise positions [17, 23, 40, 46, 102], e.g., GPS satellites [70] used by GPS enabled drones for outdoor light shows. In Figures 1.d and 1.e, an FLS lacks line of sight with a GPS satellite. It uses those FLSs with a high confidence in their location as its anchor. The position of these anchors is not 100% precise. This causes triangulation and trilateration to produce distorted shapes.

Assumptions of the SwarMer framework about its FLSs are as follows. First, FLSs are cooperative. Second, an FLS may adjust its radio range by manipulating its transmission power to communicate with FLSs that are farther away [92]. Third, each FLS has a unique id, FID. FID may be assigned by the FLS hardware, e.g., MAC address of its networking card. Alternatively, a dispatcher [31] may assign an FID to an FLS as it deploys FLSs. Fourth, an FLS knows its assigned coordinate (point in a point cloud) and the coordinates of its neighbors. The exact number of known neighbors is a tradeoff between the amount of storage required from each FLS and the required communication bandwidth for the FLSs to communicate with their neighbors and obtain this information from them.

Contributions of this paper are:

- SwarMer as a relative localization framework to render 2D and 3D shapes. A localization technique is a plugin for the SwarMer framework [3]. This decentralized framework uses swarm concepts. FLSs may execute SwarMer continuously. To the best of our knowledge SwarMer is novel and has not appeared elsewhere. (See Section 3.)

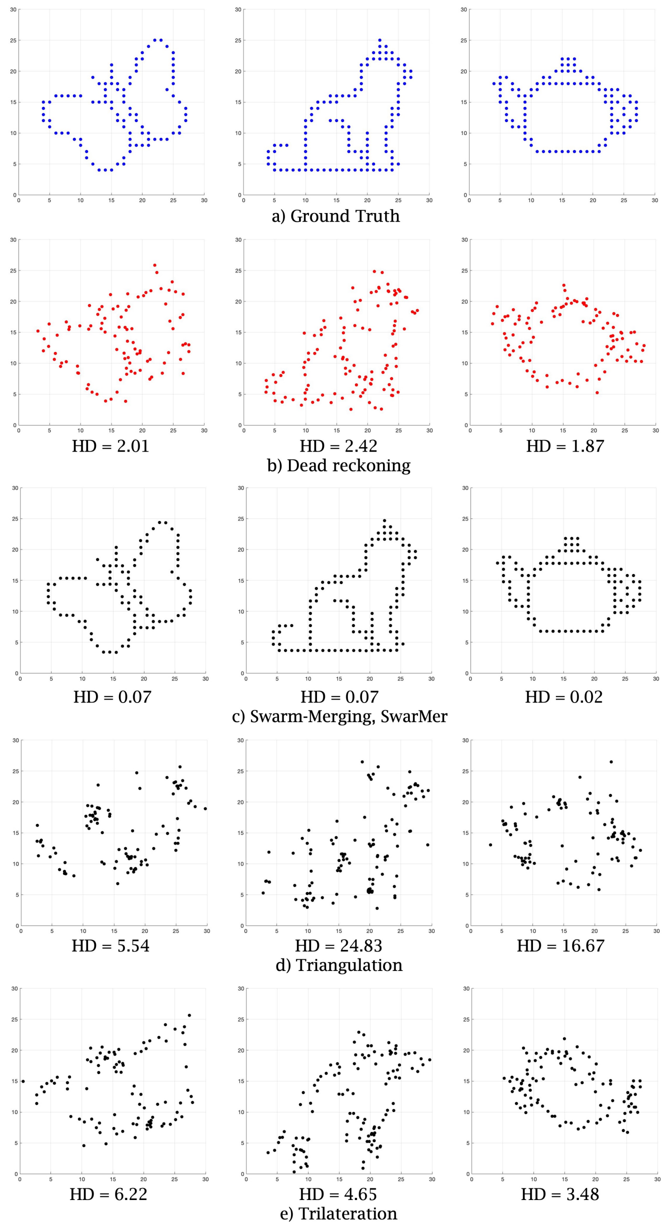


Figure 1: (a) A 2D butterfly, cat, and teapot, (b) illuminations with dead reckoning using ϵ=5 degrees of error, (c) adjustments with swarm-merging using dead reckoning with ϵ=5 degrees of error, (d-e) adjustment with Triangulation and Trilateration using 100% accurate distance and angle.

- We present a MATLAB simulation and a Python 3.9 implementation of SwarMer. We use the simulator to obtain insights into the alternative design decisions. We also use it to implement triangulation and trilateration[1] as the base

[1]Both triangulation and trilateration may serve as a localization plugin of the SwarMer framework. We do not pursue them in this role because a technique of [3] is simpler and highly accurate.

comparison techniques with SwarMer. The emulator implements the most promising designs with adjustable configuration parameters. Its processes use UDP sockets and we are actively exploring their deployment in a swarm of drones. (See Sections 5 and 6.)

- We open source our software implementations, and the 2D and 3D data used in this study for further use by the scientific community. See https://github.com/flslab/SwarMer/ for details.
- A limitation of SwarMer is that it may shift a shape along different axes. It requires an accurate reference point or an *oracle* FLS to address this limitation. (See Section 4.2.2.)

The rest of this paper is organized as follows. Section 2 provides the terminology used by this paper and a formal statement of the localization problem. Section 3 presents SwarMer. We evaluate SwarMer using 2D and 3D point clouds in Section 4 and compare it with triangulation and trilateration in Section 5. Section 6 presents an implementation of SwarMer. Related work are described in Section 7. Section 8 presents our future research directions.

2 TERMINOLOGY, PROBLEM STATEMENT

Given the novelty of FLSs and a 3D FLS display, this section introduces terminology for concepts in order to state the problem solved by SwarMer. We use this terminology throughout the paper. The SwarMer framework is independent of the assumed terminology and may use alternative ontologies that match the concepts per definitions provided in this section.

To render an illumination, an FLS display constructs a 3D mesh on the display volume. A cell of this mesh, a display cell [31], is dictated by the downwash of an FLS [5, 8, 27, 63, 99]. Assuming an FLS is a quadrotar, a cell may be an ellipsoid [5, 8, 63] or a cylinders [27, 99] that results in a larger separation along the height dimension. Each display cell has a unique Length, Height, and Depth (L,H,D) coordinate [31]. We use the L, H, D coordinate system instead of X, Y, Z to identify a cell because there is no consensus on one definition of the Y and Z axes. While the picture industry uses the Z axis as the depth, mathematicians use the Y axis for the depth. It is trivial to map our L, H, D coordinate system to either definition without ambiguity.

We assume a fixed coordinate system for the FLS display with (0, 0, 0) as its bottom left corner. Moreover, we assume the geometry of a 3D point cloud identifies the location of a display cell in an illumination cell. This is the (L,H,D) coordinate of a point assigned to an FLS to illuminate.

Definition 2.1. *Ground truth* is the ideal location and orientation of an FLS in an FLS display. It is dictated by the geometry of a point cloud, the (L,H,D) coordinates of a point. An FLS is assigned the coordinates of a point in the ground truth to illuminate.

Definition 2.2. *Estimated truth* is the actual location and orientation of an FLS in an FLS display. It is an FLS's estimate of the ground truth and may match the ground truth with some degree of accuracy including perfect (100%) accuracy.

Definition 2.3. A *vector* $\vec{i,j}$ exists between any two points i and j in a point cloud. It has a fixed length $|\vec{i,j}|$. With 2D point clouds, it has an angle θ relative to the horizontal L-axis. With 3D point clouds, in addition to θ, it has an angle ϕ relative to the height, H-axis, of a display.

PROBLEM DEFINITION 1. *Given a 2D (3D) point cloud with each point assigned to a unique FLS, localize FLSs to minimize the Hausdorff distance [38] between the estimated truth and the ground truth.*

We use the Hausdorff distance [38] to compare the FLS rendering E, i.e., the estimated truth, with the ground truth G, Hausdorff(E,G) or HD for short. This metric reports the maximum error in distance between E and G after applying a translation. A lower value is better with zero reflecting a perfect match between E and G.

The translation process is required because SwarMer is a relative localization technique. With the emulator of Section 6, this process computes the center[2] of each point cloud E and G, C_E and C_G, respectively. It computes a vector $\vec{D} = C_G - C_E$. Next, it adds $\vec{D}$ to each point in the point cloud E. Finally, we compute HD.

3 SWARM-MERGING (SWARMER)

SwarMer is a decentralized algorithm that assumes the error in dead reckoning decreases when FLSs travel shorter distances. A dispatched FLS travels to its assigned destination using dead reckoning. Once at its destination, SwarMer requires the FLS to localize relative to its neighbors using dead reckoning repeatedly. This reduces the travelled distance to enhance the accuracy of dead reckoning, enabling the FLS to approximate the ground truth more accurately.

As soon as an FLS F_i arrives at its destination, it constructs a swarm consisting of itself with its Swarm-ID set to its FID, ξ_i=i. Next, F_i expands its radio range to identify an FLS F_j with a different swarm-id, $\xi_i \neq \xi_j$. F_i challenges F_j to merge into one swarm. Assuming F_j accepts the challenge, the FLS with a lower Swarm-ID becomes the *anchor* FLS and the other FLS becomes the *localizing* FLS. Assuming $\xi_i < \xi_j$, the localizing FLS, F_j, computes a vector $\vec{V}$ for its movement relative to its anchor F_i to better approximate the *ground truth.* Two techniques to compute $\vec{V}$ are presented at the end of this section.

An FLS maintains a status property initialized to `available`. Its other possible value is `busy`. When F_j is challenged and its status is `available` then it changes its status to `busy` and notifies the FLSs that constitute its swarm to change their status to `busy`. F_i also changes its status to `busy` and informs those FLSs in its swarm to change their status to `busy`. Each message includes whether the swarm is localizing or anchor. F_i and F_j determine this independently, by comparing their Swarm-ID, ξ_i and ξ_j. With $\xi_i < \xi_j$, in addition to the `busy` status change, F_i's (F_j's) message includes the anchor (localizing) flag.

An `available` FLS accepts a challenge always. A `busy` FLS accepts a challenge only if it is an anchor and its Swarm-ID is lower than that of a challenger. This FLS will serve as the anchor for the challenger. An FLS may serve as the anchor for multiple localizing FLSs beloning to different swarms. Moreover, different FLSs belonging to the same swarm may serve as anchors for multiple localizing FLSs belonging to different swarms. This enables $M \geq 2$ swarms to merge into one.

Once F_j computes a vector $\vec{V}$ for its movement relative to F_i, F_j informs members of its swarm to change their Swarm-ID to

[2] The center of a point cloud is the average of the coordinates of its points.

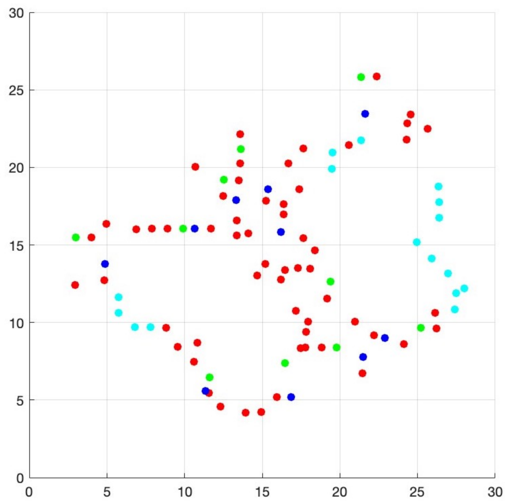


Figure 2: FLSs of different swarms localize concurrently.

F_i's Swarm-ID (ξ_i) and move along $\vec{V}$. F_j changes its Swarm-ID to F_i's Swarm-ID (ξ_i) and moves along $\vec{V}$ to localize itself relative to F_i. Next, it changes its status from `busy` to `available`. F_j's local movement and status change may occur concurrently with its message transmission to the members of its swarm. Members of F_j's swarm change their Swarm-ID to ξ_i, move along $\vec{V}$, and change their status to `available` independently. Finally, F_j notifies F_i to unanchor itself, enabling it to localize relative to other FLSs. This completes merging of the F_i and F_j swarms into one with the participating FLS's Swarm-ID set to ξ_i.

SwarMer localizes FLSs using their nearest neighbor by requiring an FLS to expand its radio range. FLSs must measure distance and angle relative to one another to localize. With radio signals such as Ultra-Wide Band (UWB), the quality of these measurements is enhanced with shorter distances [14]. If such a limitation does not exist, an FLS may localize using an arbitrary neighbor.

Definition 3.1. A *swarm* consists of a collection of FLSs with the same Swarm-ID ξ. ξ is set to the minimum FID of the FLSs in the swarm. FLSs that constitute a swarm move together [67, 88].

FLSs belonging to different swarms may localize concurrently. One or more FLSs of a swarm may serve as the anchor for different localizing FLSs. Figure 2 shows 10 localizing FLSs (green), their anchored FLSs (dark blue), members of their swarms (red), and available FLSs (light blue).

When an FLS expands its radio range, it may encounter $M - 1$ FLSs with different Swarm-IDs. The FLS with the lowest Swarm-ID becomes the anchor while $M - 1$ FLSs localize relative to it. This merges M swarms into one. SwarMer may limit the value of M to 2 or higher (∞), e.g., 2 for binary merging of swarms[3]. The value ∞ merges as many candidates as possible.

SwarMer may use different techniques to select the identity of the anchor swarm. Thus far, we have described the *Lowest Swarm-ID*, a simple technique that uses the FLS with the lowest Swarm-ID as the anchor. A more complex technique is to use the FLS belonging to the swarm with the most number of FLSs as the anchor FLS. It may require an FLS to use a decentralized peer-to-peer technique to estimate the size of its swarm [9]. In case of ties with multiple swarms having the same size, the one with the lowest Swarm-ID may serve as the anchor. In [3], we evaluate these four techniques. Obtained results show while one technique may be superior for a point cloud, no technique is superior across all point clouds. Hence, we use the lowest Swarm-ID for the rest of this paper.

With F FLSs, the number of rounds performed by SwarMer to construct one swarm is $log\ F$. With M=2, the number of rounds is deterministic. With M=∞, the base of the log is a function of the average number of swarms merged in a round. It depends on the density of the point cloud and how many swarms are merged at a time. See [3].

Multiple FLSs, say μ, of a swarm may race with one another to localize relative to different FLSs belonging to different anchor swarms concurrently. Thus, μ localizing FLSs generate different vectors for the FLSs of the localizing swarm. SwarMer ensures an FLS of the localizing swarm applies at most one vector, discarding $\mu - 1$ vectors. This partitions the localizing swarm into μ swarms. These swarms will subsequently localize relative to one another to form one swarm. See [3] for details of this race condition.

As swarms merge and form larger swarms, SwarMer saves battery power of FLSs by requiring only those FLSs with a missing neighbor to expand their radio range. This saves battery power of many FLSs by preventing them from expanding their radio range only to discover FLSs that decline their challenge to merge. FLSs with no missing neighbors are termed Relatively-Complete, *R-Complete*. They wait to be challenged.

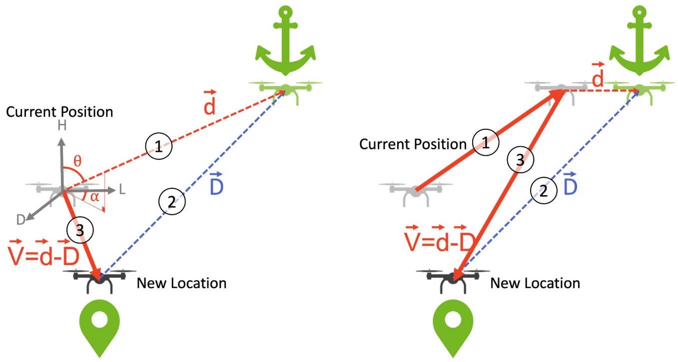
SwarMer imposes the following requirement on all FLSs. A swarm consisting of an FLS with busy neighbors belonging to a different swarm is required to localize itself relative to one of the neighbors and join its swarm. This makes SwarMer resilient to FLS failures and FLSs leaving to charge their battery. In both cases, a new FLS arrives to substitute for the departing FLS. The requirement causes this new FLS to localize and join the existing swarm immediately. In addition, this requirement prevents formation of multiple disjoint swarms that should be one with small values of η.

One may configure SwarMer to localize FLSs continuously. Once all FLSs constitute one swarm, they may thaw their swarm membership. Each FLS resets its Swarm-ID to its FID, Swarm-ID=FID. This forms swarms consisting of one FLS without impacting the position of each FLS. Next, the FLSs repeat the localization process of SwarMer. Sections 4.2 and 6 show repeated application of SwarMer enhances the quality of a rendering, i.e., lowers the Hausdorff distance. We describe several techniques to thaw the final swarm in a decentralized manner [3]. One is detailed in Section 6.

Relative Localization: This section describes two techniques for a localizing FLS to position itself relative to an anchor FLS. They are plugins for the SwarMer framework and may be implemented using a variety of techniques described below.

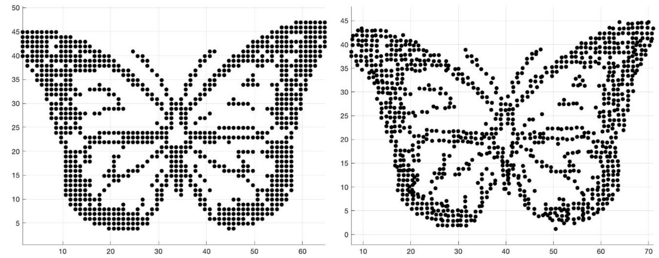
The first technique, Signal Strength (SS), assumes an FLS (a) has sensors to detect both the angle and the strength of the signal received from another FLS, and (b) may convert these sensor readings to distance from the FLS generating the signal. One may consider and evaluate alternative sensors when designing

[3] M=∞ renders a shape accurately faster when compared with M=2 [3]



a) Signal Strength b) Physical Movement

Figure 3: Two localization plug-in techniques for SwarMer.



a) Signal Strength, HD = 0.00 b) Physical Movement, HD = 1.38

Figure 4: A comparison of SS with PM, ϵ=5°.

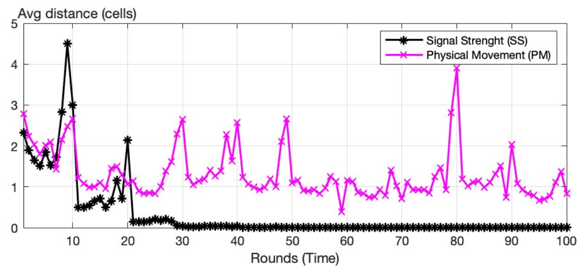


Figure 5: Dead reckoning distance for the 2D Butterfly of Figure 4.

and implementing SS with FLSs. Bluetooth [58, 89], Wi-Fi [65, 69], RFID [11, 59, 93, 105], UWB [45, 50, 74], Lidar [21, 75], RGB cameras [10, 60, 75, 103] and infrared [44, 63, 68] may be used to obtain distance information. The angle information may be obtained by mounting an antenna array on an FLS, e.g., Bluetooth [86], Wi-Fi [79]. Angle of Arrival, AOA [96], obtains the angles at which the signal sent by a localizing FLS reaches the antennas mounted on an anchor FLS. Several surveys document the alternative sensor technologies, their strengths and weaknesses [18, 35, 51, 101].

SS requires the localizing FLS to request the anchor FLS to generate a signal. It uses its sensors to estimate vector $\vec{d}$. This vector is defined using angle α relative to the L-axis with distance |d| from the anchor FLS, Step 1 in Figure 3.a. SS also computes the angle θ relative to the height H. Next, the localizing FLS uses the ground truth to calculate vector $\vec{D}$. The tail (head) of $\vec{D}$ is the position of the localizing (anchor) FLS in the ground truth, Step 2 in Figure 3.a. We subtract the two vectors to obtain $\vec{V}$, $\vec{V} = \vec{d} - \vec{D}$. The localizing FLS moves along vector $\vec{V}$ to its new position, Step 3 in Figure 3.a.

The second technique, named Physical Movement (PM), requires the localizing FLS to position itself next to the anchor FLS using its IMU. Next, the localizing FLS computes its angle and distance from the anchor FLS in the ground truth, vector $\vec{D}$ of Step 2 in Figure 3.b. It also computes its displacement vector relative to the anchor, Vector $\vec{d}$. We subtract the two vectors to obtain $\vec{V}$, $\vec{V} = \vec{d} - \vec{D}$. The localizing FLS moves along vector $\vec{V}$ to its new position, see Step 3 in Figure 3.b.

Generally, the dead reckoning distance travelled by a localizing FLS is greater with PM when compared with SS, see Figure 5. The Hausdorff distance decreases close to zero with SS because the dead reckoning distance is decreasing as a function of the number of iterations. PM does not realize the same. Thus, SS provides more accurate renderings. This is especially true with lines, compare Figure 4.a with Figure 4.b.

In summary, SS is both faster and more accurate than PM. It is also more energy efficient by minimizing the distance traveled by FLSs. This maximizes the flight time of an FLS on a fully charged battery. These observations hold true with both 2D and 3D shapes.

FLS Failure and Battery Charging: An FLS is a mechanical device that may fail. One may use the concept of leases [32, 33] to prevent failure of a localizing FLS from causing another FLS to remain anchored indefinitely. When a localizing FLS requests another FLS to serve as its anchor, the anchored FLS grants a lease with a fixed duration δ to the localizing FLS. The localizing FLS must renew this lease prior to its expiration, i.e., δ time units. Otherwise, the anchored FLS will assume the localizing FLS has failed and will un-anchor itself, i.e., change its status to `available`.

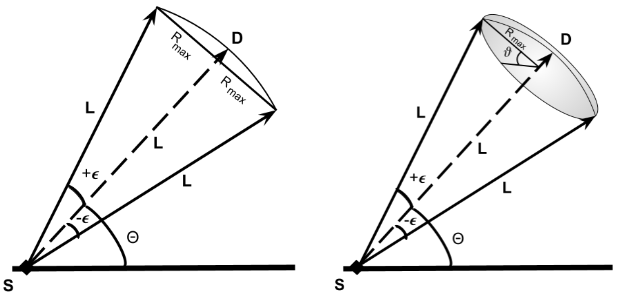
When an anchor FLS fails, as long as the localizing FLS has sufficient information to localize then it localizes, informs the members of its swarm to update their Swarm-ID to that of the anchor FLS, and changes its status to `available`. SwarMer uses leases to detect failures in an asymmetric manner: Failure of the anchor does not impact the localizing FLS and leases do not detect a failed anchor FLS. SwarMer uses leases to detect failure of the localizing FLS, causing the anchor FLS to unanchor itself.

A standby FLS [31] will substitute for a failed FLS as follows. It uses dead reckoning to move to the location of the failed FLS and localizes itself relative to a neighbor.

When the battery flight time of an FLS is below a threshold, it notifies its neighbors and the standby that it is leaving the swarm. While it flies back to a charging station, a standby FLS assumes its lighting responsibility per the discussion of failure handling.

4 AN EVALUATION

We simulate SwarMer using MATLAB and evaluate it using several 2D and 3D point clouds, see Figures 1.a and 7.a. The simulator assigns a point of a point cloud to an FLS and implements dead reckoning to compute the location of the FLS deployed from a dispatcher in its estimated truth. This dead reckoning technique is



(a) 2D. **(b) 3D.**

Figure 6: Angle of error ϵ with dead reckoning.

described in Section 4.1. SwarMer uses this technique repeatedly to localize FLSs.

The simulator executes in rounds. At the start of each round, an FLS executes the algorithm of Section 3. We quantify (a) the minimum, average, and maximum number of FLSs that SwarMer localizes in each round, (b) the minimum, average, and maximum distance traveled by FLSs during each round, (c) the number of swarms at the end of each round, (d) the Hausdorff distance at the end of each round.

This evaluation uses standard PNG images for 2D point clouds and the Princeton Shape Benchmark [72] for 3D point clouds. We wrote software to convert both into point clouds, see Figure 1.a.

4.1 Dead reckoning

An FLS has a starting position S and a destination D in the ground truth, see Figure 6. Vector $\vec{SD}$ at angle θ has a fixed length L. The error in dead reckoning is bounded by the angle ϵ, $\theta \pm \epsilon$. In 2D (3D), see Figure 6.a (6.b), a radius R_{max} for the circular-arc (spherical dome) defined by ϵ. R_{max} increases as a function of ϵ and L.

In 3D, in addition to ϵ, another random number ϑ between 0 and 2π radian (i.e., 0 and 360 degrees) is used to decide the direction in which the epsilon should be added to the vector. Figure 6.b shows ϑ in the spherical dome.

The value of ϵ is a configuration parameter of the simulator and defines R_{max} in Figure 6. Every time an FLS invokes dead reckoning, the simulator generates a random value between $\pm\epsilon$. It adds this angle to θ to compute a vector with length L starting at S. The point D at the end of this vector is the destination of an FLS and its coordinate in the *estimated truth*. The FLS flies to this coordinate.

4.2 Experimental Results

Figure 1 shows SwarMer localizes FLSs to approximate the ground truth with ϵ=5°, compare Figure 1.c with Figure 1.a. These 2D point clouds consist of approximately 100 FLSs. SwarMer employs dead reckoning with the Signal Strength (SS) technique and η=5. It reduces error by minimizing the distance traveled by localizing FLSs.

Figure 7 shows several 3D point clouds from the Princeton Benchmark that challenged our earlier attempts at developing a localizing algorithm. The dragon, hat, and skateboard[4] consist of 760, 1562, and 1727 points (FLSs), respectively. Figure 7.b shows dead reckoning by itself results in distorted renderings. These are enhanced by SwarMer, resulting in a low HD, see Figure 7.c. In these experiments, SwarMer merges as many as 6 swarms in a round. SwarMer is a continuous framework. In our simulation studies, we artificially terminated the simulation once there was a single swarm with an HD lower than 0.09. This is realized in rounds 21, 16, and 17 with the dragon, the skateboard, and the hat, respectively.

Figure 8 shows SwarMer's HD with different degrees of error as a function of the number of rounds. A lower HD is better because it shows the FLS rendering approximates the ground truth more accurately. This distance decreases as a function of the number of rounds. With ϵ=1°, the simulation terminates in the 5th round because its HD drops below 0.09.

[4]Their Princeton shape files are m1625, m1630, and m1619 respectively.

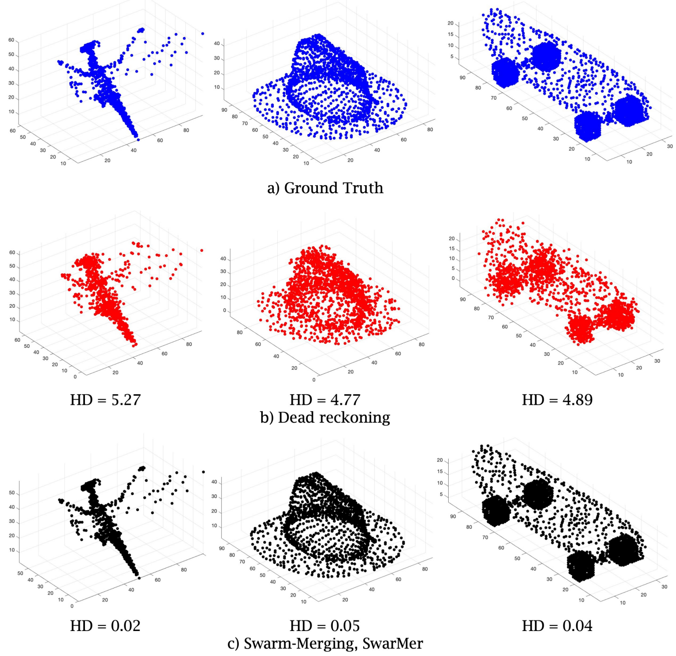


Figure 7: (a) Ground truth, dead reckoning, and SwarMer for a dragon, a hat, and a skateboard. ϵ=5°, a random ϑ, $0 \leq \vartheta < 360$.

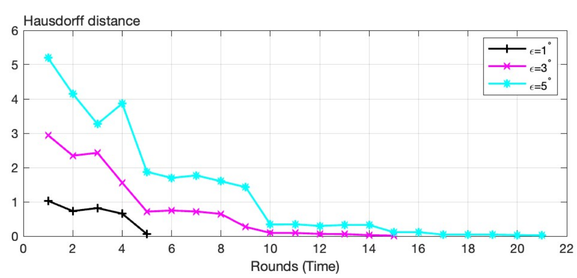


Figure 8: HD of the dragon with different ϵ.

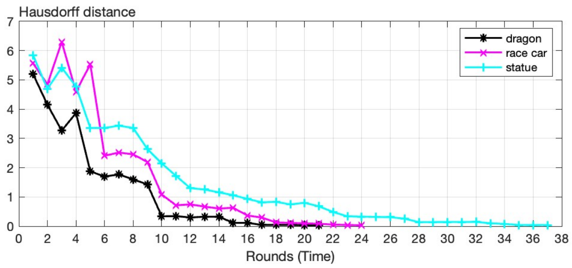


Figure 9: Shapes with tens of thousands of FLSs, ϵ=5°, M=∞.

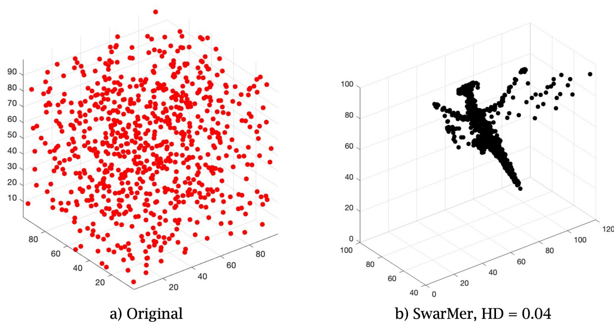
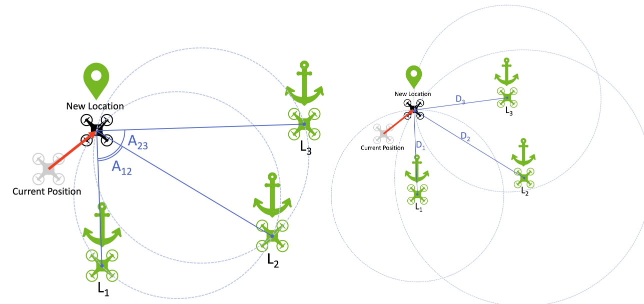


Figure 10: Random placement, https://youtu.be/cZrz0e61txU.

4.2.1 Scalability. SwarMer is decentralized and scales to render shapes consisting of many FLSs accurately and quickly. We evaluated SwarMer using several large point clouds produced using the Princeton shape benchmark [72]. We report on two. First, m1510, a race car, consisting of 11,834 unique points. Second, m303, a statue of a face, consisting of 22,310 unique points. With both, the number of rounds required to construct a single swarm is 6. As a comparison, it is 5 with the dragon consisting of 760 points. However, the HD of the race car and the statue is approximately twice worse than the dragon in the sixth round. See Figure 9. The number of rounds to realize a HD<0.09 with ϵ=5° is 24 and 37 for the race car and the statue, respectively. It is 21 rounds for the dragon. However, the race car has more than 10x points and the statue has more than 20x points when compared with the dragon.

4.2.2 Random FLS placement. SwarMer works with random placement of FLSs in a 3D display as long as the coordinates of a point are assigned to each FLS. A limitation is that the shape may be shifted along the different axes by a wide margin because SwarMer is a relative localization technique. See Figure 10.b. The precise amount of shift depends on how FLSs form swarms, and how swarms merge and move. One may address this limitation by having either a reference point on the display with a well known position or an *oracle* FLS configured with specialized hardware that computes its position with a high accuracy. A swarm that localizes relative to either inherits the oracle property. All other swarms are required to localize relative to its FLSs, preventing a shape from shifting across different axes.



(a) Triangulation. **(b) Trilateration.**

Figure 11: Triangulation and Trilateration.

5 A COMPARISON

In our first approach to localize FLSs to render shapes, we considered the use of triangulation and trilateration. We hypothesized that using those FLSs with a high confidence in their location as anchors will produce shapes with a high accuracy. This hypothesis proved wrong because trilateration and triangulation compound small errors in anchor locations to produce distorted shapes. Below, we describe how an FLS computes its confidence. Subsequently, we describe triangulation and trilateration in turn. This section ends with a comparison of these techniques with SwarMer.

Confidence: Once an FLS i (F_i) arrives at its destination, it may compute its confidence C_i using either the average or worst estimated error attributed to dead-reckoning. With average (worst), it computes the average (maximum) radius R around the semi-circle in 2D (semi-sphere in 3D) as the error in its position relative to each of its n_i neighbors. Its confidence is defined as:

$$C_i = 1 - \sum_{k=1}^{n_i} min(\frac{1}{n_i}, \frac{R}{\delta_{gt}(F_i, F_k)}) \quad (1)$$

where $\delta_{gt}(F_i, F_k)$ is the distance between FLS i and a neighboring FLS k in the ground truth. When a neighbor is missing, R is set to a large value, e.g., max integer. This causes Equation 1 to use $\frac{1}{n_i}$ for the missing neighbor. A large value of C_i means the FLS's estimated truth reflects the ground truth more accurately. C_i equal to 1.0 is ideal.

Three Object Triangulation requires a localizing object to compute its location and orientation using its angles relative to three objects. Alternative techniques include iterative search, geometric circle intersection, and Newton-Raphson iterative method [22]. We use the geometric circle intersection method since it is widely used in the literature and ranked as the best when compared with the other two techniques [22].

The geometric circle intersection method is fast and its failures are easy to detect. It requires a localizing FLS, the one with the lowest confidence relative to its neighbors, to compute two circles using the estimated location and the ground truth angle of three FLSs. See Figure 11a. The three FLSs may be its neighboring FLSs. In most cases, the two circles intersect at two points. These points are the mutual FLSs and the location of the localizing FLS. The latter has an angle relative to the other three FLSs in the ground-truth. If these angles match those in the estimated truth then the localizing

FLS does not move. Otherwise, the localizing FLS computes a new location for itself that matches the angles in the ground truth. The localizing FLS flies to this new location.

This method fails when the center of the two circles is too close (we used 1 illumination cell) or when two circles overlap.

Three Object Trilateration is the use of distances or ranges for determining the unknown location of an object. The localizing FLS i computes its distance to three neighboring FLSs in the ground truth: D_1, D_2, and D_3. See Figure 11b. Each delta is the radius of a circle with the corresponding neighbor as its center. FLS i computes a new estimated location that matches D_1, D_2, and D_3 in its estimated truth. This is the new location of FLS i.

Trilateration fails when either the three circles do not intersect at the same point in the ground truth, or there is no location in estimated truth that matches D_1, D_2, and D_3. While rare, both do occur in our experiments.

Our implementation of both triangulation and trilateration assumes Signal Strength (SS) measurements in the estimated truth match those of the ground truth with 100% accuracy. It also assumes an FLS is able to measure angle with triangulation and distances with trilateration with 100% accuracy. We made these assumptions because triangulation and trilateration perform poorly. We wanted to remove error in measuring angle and distance as a possible explanation for their inferior localization.

Experimental Results Figure 1.b shows rendering of several 2D shapes using dead reckoning with ϵ=5°. Figure 1.c shows application of SwarMer to each shape. It continues to use the same ϵ when applying dead reckoning using signal strength. The last two rows of Figure 1 show triangulation and trilateration do not improve the quality of illumination. With both, an FLS uses the worst estimated error attributed to dead-reckoning to compute its confidence. This is despite the fact that both triangulation and trilateration are provided with 100% accurate angle and distance measurements.

The number of points in a point cloud impacts the quality of renderings with small values of ϵ. Figure 12 shows a 2D butterfly consisting of 1008 points, more than ten times the number of points in the butterfuly of Figure 1. The FLSs deployed from the origin, (0,0), travel the longest distance to the rightmost wing of the butterfly, incurring the highest amount of error and the most distortion with ϵ=5°. With ϵ=1°, the shape of the butterfly is recognizable with both triangulation and trilateration. However, their Hausdorff distance is worse than dead reckoning by itself. With ϵ=3 and 5 degrees, triangulation results in a few FLSs to move away from the illumination by a large distance. This is reflected in their longer scale of the x-axis in Figures 12c. While trilateration does not suffer from this limitation, its localization results in distorted shapes with a large Hausdorff distance. SwarMer continues to provide accurate localization subjectively, see Figures 12b. It realizes a Hausdorff distance lower than 0.09 by executing 20, 30, and 40 rounds with ϵ=1, 3, and 5 degrees, respectively.

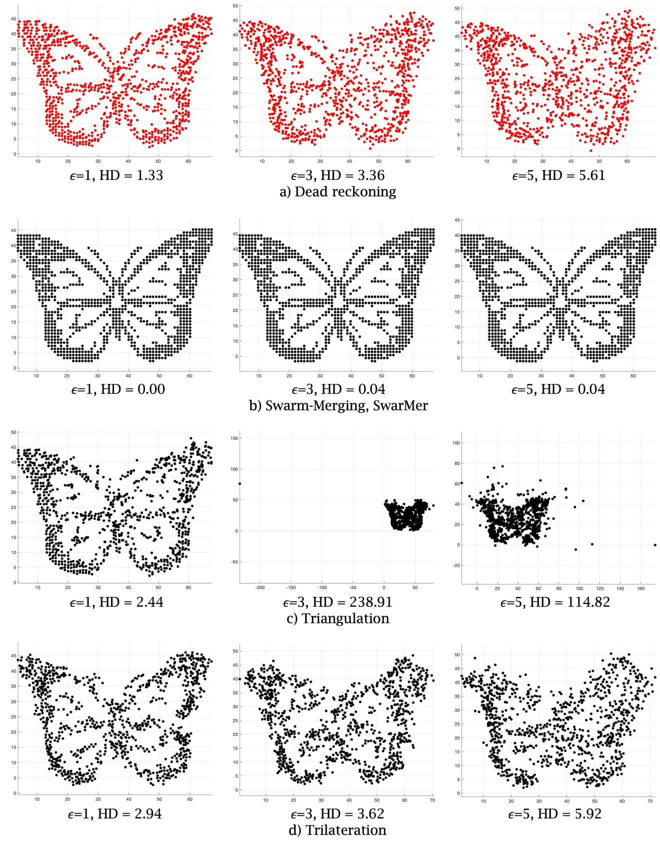


Figure 12: (a) A 2D butterfly with dead reckoning and three different degrees of error, (b) with SwarMer, (c) with Triangulation, and (d) Trilateration..

6 AN IMPLEMENTATION

We implemented SwarMer as a Pythond 3.9 process (P_{FLS}) to execute on board an FLS. Processes deployed on different FLSs exchange messages to implement SwarMer. We used a cluster of Amazon AWS instance u-6tb1.112xlarge to conduct the experiments reported in this section. Each AWS instance consists of 448 virtual cores and 6 Terabytes of memory. While the Dragon required a cluster of 6 servers, the Hat and the Skateboard each used a cluster of 16 such servers because they consist of more points (FLSs).

This implementation is different from the Matlab simulation model in several ways. First, each P_{FLS} implements the concept of time by either sleeping or busy-waiting for the duration of time it travels from its current location to a new destination. It has a realistic velocity model that controls its acceleration and deceleration to arrive at its destination with a speed of zero. Second, each P_{FLS} implements a UDP-Wrapper that emulates a networking card with an adjustable radio range. The time required for a P_{FLS} to expand its radio range and transmit messages is another difference from the Matlab simulator. Third, a P_{FLS} may incur different forms of failures such as packet loss, and a process shutdown to emulate FLS failures. Finally, a P_{FLS} uses a probabilistic method to thaw the final swarm. With this method, a P_{FLS} sets a timer to expire between H and $2H$ seconds where $H = log_2 F$ and F is the number of FLSs. Once an FLS's timer expires, it broadcasts a thaw message using its maximum radio range. A receiving FLS sets its Swarm-ID to its FID and resets its timer.

A P_{FLS} implements a finite state machine that maintains the following 5 states: Deploying, Available, Busy Anchor, Busy Localizing, and Waiting. 11 events corresponding to 11 different message types transition the state of a P_{FLS} to implement SwarMer.

A P_{FLS} consists of two threads: A networking thread ($T_{Network}$) and a state mutation thread (T_{Mutate}). While $T_{Network}$ only reads the FLS state, T_{Mutate} may both read and write the FLS state. $T_{Network}$ blocks on the network socket and processes messages as fast as they arrive by reading the current state of the machine. One of its objectives is to release the lease granted to an FLS that has decided to become an anchor FLS itself. This happens due to an undesirable race condition[5]. In addition, it is designed to minimize latency between cooperating FLSs, expediting the execution of SwarMer to form shapes faster. $T_{Network}$ places those messages that change system state in a queue for processing by T_{Mutate}. T_{Mutate} consumes these messages from the queue and may update the state of the machine to transition its state. The queue introduces delays and $T_{Network}$ is designed to minimize the impact of this delay on a swarm of cooperating FLSs.

In our experiments with drones, we have observed the communication link to drop packets. A P_{FLS} uses UDP (instead of TCP) to communicate with other P_{FLS}s. UDP is an unreliable communication protocol. It may drop packets, deliver a sequence of packets transmitted by a P_{FLS} out of order, and delay the delivery of packets. Our implementation of SwarMer is non-blocking to accommodate the first two features of UDP. With UDP's last two features, we use a monotonically increasing message id for each packet sent by a P_{FLS_i}. A P_{FLS_j} maintains the largest message id it has received from another P_{FLS_i} and may drop those with a smaller id.

The lessons learned from this implementation are as follows. First, SwarMer is fast, reducing the HD by several orders of magnitude in a few seconds. Second, SwarMer tolerates a high packet loss rate to construct one swarm. Hence, it continues to reduce the HD significantly in the first few seconds. Third, while FLS speed impacts the initial deployment of FLSs using dead reckoning, it does not impact how fast SwarMer improves the quality of an illumination (i.e., drop in HD) as a function of time. The key insight is that the average distance moved by an FLS (or swarm) to localize is small. This prevents an FLS from accelerating to its maximum speed. Fourth, SwarMer is resilient to FLS failures. It incorporates FLSs that replace failed FLSs seamlessly. Finally, FLS latency in processing messages may result in undesirable race conditions that slow-down SwarMer without impacting its functionality. This explains our use of $T_{Network}$ thread to minimize latency.

Obtained Results. Figure 14 shows the HD as a function of time for the different shapes in Figure 7. The y-axis of this figure is linear scale. It reports HD as a function of the length of a display cell, a cube with a 5 Centimeter length. Hence, a HD of 1 means the maximum distance between the points in the ground truth and the estimated truth is 5 Centimeter. HD drops significantly in the first few seconds of SwarMer's execution as FLSs localize relative to one another. When this distance drops below 50 Micrometer (corresponding to HD of 0.001), we consider it close to zero and stop reporting the HD. The different shapes arrive at this threshold at different times. See the variant of the Figure inside its empty space with y-axis in log scale. The Dragon arrives at this threshold soonest as it consists of 760 FLSs. The Skateboard requires the longest as it consists of more than twice (1727) as many FLSs.

Figure 13 shows the 3D shapes with different HD values. There is no noticeable difference between the ground truth and the estimated truth with HD values lower than 0.1, a 5 millimeter error. See the anonymized video links for a demonstration.

We designed the UDP-Wrapper to support both symmetric and asymmetric packet loss. It may be configured to drop packets with a pre-specified probability either at the transmitting, receiving, or both. In our experiments, SwarMer was resilient to these errors. Depending on the lost event, SwarMer may require a longer time to construct one swarm. However, it continues to decrease the HD dramatically in the first few seconds of its execution. To illustrate, Figure 15 shows HD with the Skateboard and different rate of packet loss. The results with asymmetric packet loss are similar.

SwarMer's use of leases enables it to tolerate FLS failures. Moreover, a replacement FLS deployed by a dispatcher may execute SwarMer to incorporate itself. Figure 16 shows the HD with the Skateboard and failure rates[6] as high as 10%. The HD is competitive with a failure rate lower than 0.01%, i.e., 1 failure every 3 hours. With higher probabilities, the anonymized videos show the dark points are illuminated using the replacement FLSs incorporated by SwarMer. One must use a failure handling technique to minimize the impact of these dark points. An example technique is to use standby FLSs [31] to maintain a high quality of illumination. This topic is an orthogonal topic to localization.

7 RELATED WORK

The SwarMer framework to localize FLSs is novel and to the best of our knowledge has not been described elsewhere.

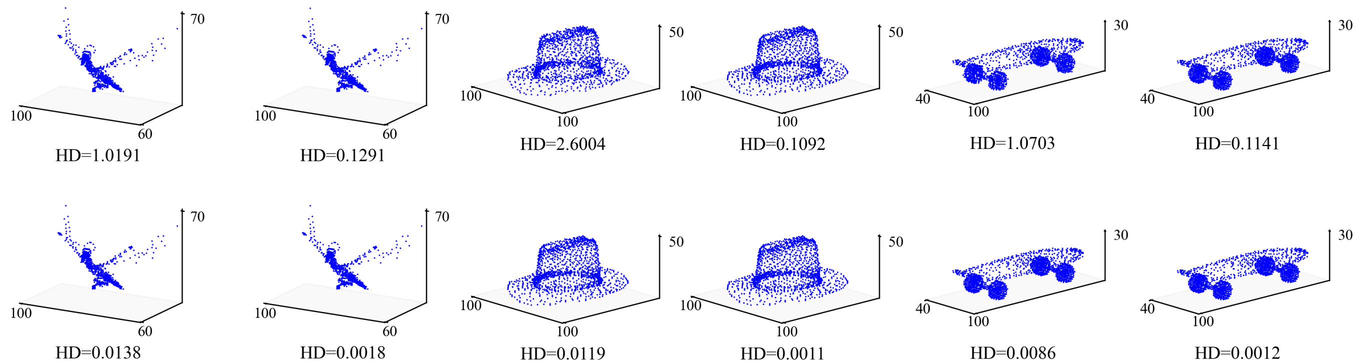
Outdoor drone shows use GPS [70] to localize and illuminate shapes in the sky. The largest show consisted of 5200 drones to celebrate the 100th anniversary of the Communist Party of China in the night sky of Longgang, Shenzhen. An FLS display is different because it does not have line of sight with GPS satellites.

The concept of a 3D display using FLSs to render multimedia shapes [2, 4, 30, 31, 61] is relatively new. Prior studies either identify localization as a challenge [2, 30, 61] or assume one exists [4, 31]. SwarMer is novel and designed for use by these studies.

Centralized indoor techniques such as optical motion capture systems merge images from fixed cameras positioned around a display volume to localize quadrotors and drones [44, 63, 68]. These systems, e.g., Vicon, are highly accurate. They require a unique marker arrangement for each drone and have enabled control and navigation of swarms of tens of drones. It may be difficult (if not impossible) to form unique markers for more than tens of small drones measuring tens of millimeters diagonally [63]. Hence their scalability is limited. Moreover, these systems require broadcast at high

[5]An example race condition is an FLS 5 challenging FLSs 4 and 6. FLS 6 accepts the challenge, causing FLS 5 to become its anchor. Concurrently, FLS 4 accepts FLS 5's challenge to become its anchor by granting a lease to FLS 5. Should FLS 5 enter the waiting state for FLS 6, its $T_{Network}$ reads Waiting state, detects the undesirable race condition, and informs FLS 4 to unanchor itself and release its lease. This minimizes the number of expired leases, expediting SwarMer.

[6]A failure of x% per FLS per second translates into an FLS failing in $\frac{100}{x}$ seconds, e.g., a 10% failure per FLS per second translates into an FLS failing in 10 seconds.



(a) Dragon, https://youtu.be/PRyjdmw8NVc **(b) Hat, https://youtu.be/y_xHdz5bs5M** **(c) Skateboard, https://youtu.be/a0fFu0z6BU0**

Figure 13: No FLS movement is visible with HD<0.1. Click the URLs for a video.

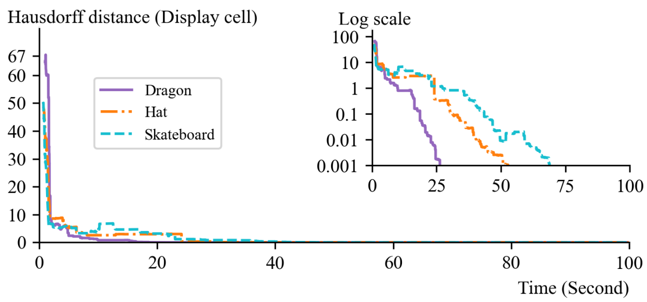


Figure 14: HD with 3D shapes of Figure 7, ϵ=5°.

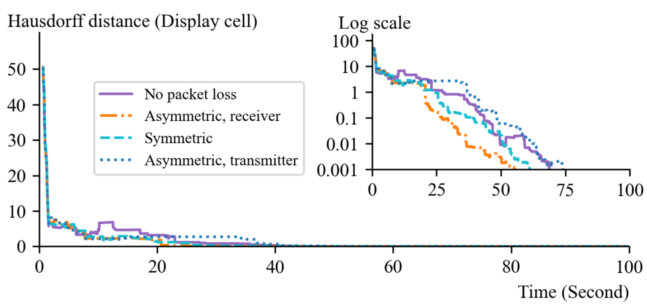


Figure 15: HD with 10% packet loss, Skateboard, ϵ=5°.

frequency from a central computer to each drone [90]. This centralized communication channel is a single point of system failure. It has a latency of several (7) milliseconds [63] and its bandwidth constrains the robustness and swarm size. SwarMer is different because it is decentralized, requiring each FLS to localize relative to its neighbor to form a swarm. The broadcast range of an FLS is controlled by its signal strength to communicate with a fixed number (η) of neighboring FLSs.

There exists a large number of studies in the area of formation control [16, 80, 85, 106], use of sensors to localize robots and drones [1, 7, 24, 25, 29, 34, 41–43, 46, 48, 54, 56, 64, 87, 94, 95, 97, 98, 100, 104, 107], and collision avoidance [5, 6, 8, 12, 15, 19, 20, 26, 27, 36, 37, 39, 44, 52, 53, 55, 62, 63, 76, 78, 81–84, 99]. These

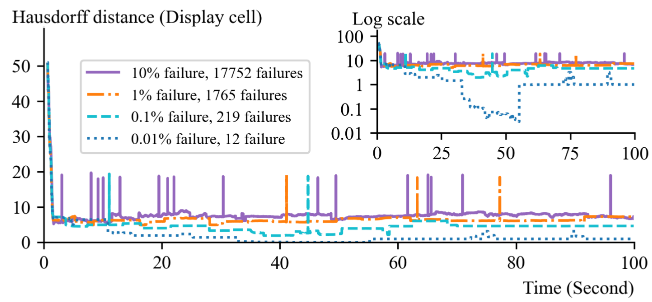


Figure 16: HD with 0.01%, 0.1%, 1%, and 10% failures per FLS per second for the Skateboard. Video clips available at https://youtu.be/YE-hzpfONwg, https://youtu.be/1Tx-_DDZf0w, https://youtu.be/QEKJgNn0Yy8, https://youtu.be/mU-oIVaNu8M, respectively.

studies complement SwarMer and may be used by a system that implements a 3D display [2]. These prior studies lack the concept of FLSs forming a group (swarm), requiring a swarm to move together, swarms merging to become one, FLSs thawing their membership to repeat the process continuously, or use leases to explicitly support FLSs failing and leaving to charge their batteries. These concepts, their decentralized design and implementation are novel and a contribution of SwarMer.

8 CONCLUSIONS AND FUTURE RESEARCH

Inspired by swarms in nature, SwarMer is a decentralized technique for relative localization of FLSs to render complex 2D and 3D shapes. It requires FLSs to localize relative to one another to form a swarm. Movement of an FLS along a vector causes its entire swarm to move along the vector. SwarMer uses the unique identifier of an FLS assigned at its deployment time to implement an organizational framework. This framework compensates for the simplicity of individual FLSs to render complex 2D and 3D shapes accurately.

We presented a MATLAB simulation and a Python implementation of SwarMer. The implementation shows SwarMer is fast. Within a few seconds, it reduces the HD to a value below 1 display cell. Its continued execution reduces the HD further to match the ground truth more accurately. It tolerates network packet loss, FLSs failing and leaving to charge their battery, and replacement FLSs arriving to illuminate points.

We are initiating an implementation of SwarMer by evaluating the relevant plugins to the SwarMer. These include an investigation of the formation control techniques, alternative sensors for a localizing FLS to orient itself relative to an anchor FLS, and a collision avoidance technique as an FLS moves along a vector, see Section 7.

9 ACKNOWLEDGMENTS

We thank Jiadong Bai and Shuqin Zhu for their valuable comments on the earlier drafts of this paper. This research was supported in part by the NSF grant IIS-2232382. We gratefully acknowledge CloudBank [57] and CloudLab [91] for the use of their resources to enable all experimental results presented in this paper.

REFERENCES

[1] Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A. Al-Ammar, and Hend S. Al-Khalifa. 2016. Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances. *Sensors* 16, 5 (2016). https://doi.org/10.3390/s16050707

[2] Hamed Alimohammadzadeh, Rohit Bernard, Yang Chen, Trung Phan, Prashant Singh, Shuqin Zhu, Heather Culbertson, and Shahram Ghandeharizadeh. 2023. Dronevision: An Experimental 3D Testbed for Flying Light Specks. https://doi.org/10.48550/arXiv.2308.10121 arXiv:2308.10121 [cs.MM]

[3] Hamed Alimohammadzadeh and Shahram Ghandeharizadeh. 2023. SwarMer: A Decentralized Localization Framework for Flying Light Specks. https://doi.org/10.48550/arXiv.2312.04571 arXiv:2312.04571 [cs.RO]

[4] Hamed Alimohammadzadeh, Daryon Mehraban, and Shahram Ghandeharizadeh. 2023. Modeling Illumination Data with Flying Light Specks. In *ACM Multimedia Systems* (Vancouver, Canada) *(MMSys '23)*. Association for Computing Machinery, New York, NY, USA, 363–368. https://doi.org/10.1145/3587819.3592544

[5] Senthil Hariharan Arul and D. Manocha. 2020. DCAD: Decentralized Collision Avoidance With Dynamics Constraints for Agile Quadrotor Swarms. *IEEE Robotics and Automation Letters* 5 (2020), 1191–1198. https://doi.org/10.1109/LRA.2020.2967281

[6] Federico Augugliaro, Angela Schoellig, and Raffaello D'Andrea. 2012. Generation of Collision-Free Trajectories for a Quadrocopter Fleet: A Sequential Convex Programming Approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems.* 1917–1922. https://doi.org/10.1109/IROS.2012.6385823

[7] Andrej Babinec, Ladislav Jurišica, Peter Hubinský, and František Duchoň. 2014. Visual Localization of Mobile Robot Using Artificial Markers. *Procedia Engineering* 96 (2014), 1–9. https://doi.org/10.1016/j.proeng.2014.12.091 Modelling of Mechanical and Mechatronic Systems.

[8] Daman Bareiss and Joran van den Berg. 2013. Reciprocal Collision Avoidance for Robots with Linear Dynamics using LQR-Obstacles. In *Proceedings - IEEE International Conference on Robotics and Automation.* 3847–3853. https://doi.org/10.1109/ICRA.2013.6631118

[9] Mayank Bawa, Hector Garcia-Molina, Aristides Gionis, and Rajeev Motwani. 2003. *Estimating Aggregates on a Peer-to-Peer Network.* Technical Report 2003-24. Stanford InfoLab. http://ilpubs.stanford.edu:8090/586/

[10] Patric Beinschob, Mark Meyer, Christoph Reinke, Valerio Digani, Cristian Secchi, and Lorenzo Sabattini. 2017. Semi-automated map creation for fast deployment of AGV fleets in modern logistics. *Robotics and Autonomous Systems* 87 (2017), 281–295. https://doi.org/10.1016/j.robot.2016.10.018

[11] Fabio Bernardini, Alice Buffi, Daniele Fontanelli, David Macii, Valerio Magnago, Mirko Marracci, Andrea Motroni, Paolo Nepa, and Bernardo Tellini. 2021. Robot-Based Indoor Positioning of UHF-RFID Tags: The SAR Method With Multiple Trajectories. *IEEE Transactions on Instrumentation and Measurement* 70 (2021), 1–15. https://doi.org/10.1109/TIM.2020.3033728

[12] Tomáš Báča, Daniel Hert, Giuseppe Loianno, Martin Saska, and Vijay Kumar. 2018. Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 6753–6760. https://doi.org/10.1109/IROS.2018.8594266

[13] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. 2021. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Transactions on Robotics* 37, 6 (2021), 1874–1890. https://doi.org/10.1109/TRO.2021.3075644

[14] Justin Cano, Gaël Pages, Eric Chaumette, and Jerome Le Ny. 2022. Optimal Localizability Criterion for Positioning with Distance-Deteriorated Relative Measurements. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 947–953. https://doi.org/10.1109/IROS47612.2022.9981718

[15] D. Cappello, S. Garcin, Z. Mao, M. Sassano, A. Paranjape, and T. Mylvaganam. 2020. A Hybrid Controller for Multi-Agent Collision Avoidance via a Differential Game Formulation. *IEEE Transactions on Control Systems Technology* PP (07 2020), 1–8. https://doi.org/10.1109/TCST.2020.3005602

[16] Nelson P. K. Chan, Bayu Jayawardhana, and Hector Garcia de Marina. 2021. Angle-Constrained Formation Control for Circular Mobile Robots. *IEEE Control Systems Letters* 5, 1 (2021), 109–114. https://doi.org/10.1109/LCSYS.2020.3000061

[17] Haige Chen and Ashutosh Dhekne. 2022. PnPLoc: UWB Based Plug and Play Indoor Localization. In *2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN).* 1–8. https://doi.org/10.1109/IPIN54987.2022.9918119

[18] Siyuan Chen, Dong Yin, and Yifeng Niu. 2022. A Survey of Robot Swarms and Relative Localization Method. *Sensors* 22, 12 (2022). https://doi.org/10.3390/s22124424

[19] Yu Fan Chen, Mark Cutler, and Jonathan How. 2015. Decoupled Multiagent Path Planning via Incremental Sequential Convex Programming. *Proceedings - IEEE International Conference on Robotics and Automation* 2015 (06 2015), 5954–5961. https://doi.org/10.1109/ICRA.2015.7140034

[20] Hui Cheng, Q. Zhu, Z. Liu, Tianye Xu, and Liang Lin. 2017. Decentralized Navigation of Multiple Agents Based on ORCA and Model Predictive Control. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), 3446–3451. https://doi.org/10.1109/IROS.2017.8206184

[21] Younghun Cho, Giseop Kim, Sangmin Lee, and Jee-Hwan Ryu. 2022. OpenStreetMap-Based LiDAR Global Localization in Urban Environment Without a Prior LiDAR Map. *IEEE Robotics and Automation Letters* 7, 2 (2022), 4999–5006. https://doi.org/10.1109/LRA.2022.3152476

[22] Charles J. Cohen and Frank V. Koss. 1993. Comprehensive Study of Three-Object Triangulation. In *Mobile Robots VII*, William J. Wolfe and Wendell H. Chun (Eds.), Vol. 1831. International Society for Optics and Photonics, SPIE, 95 – 106. https://doi.org/10.1117/12.143782

[23] Pablo Corbalán, Gian Pietro Picco, and Sameera Palipana. 2019. Chorus: UWB Concurrent Transmissions for GPS-like Passive Localization of Countless Targets. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks* (Montreal, Quebec, Canada) *(IPSN '19)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/3302506.3310395

[24] Renaud Dube, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. 2017. An online multi-robot SLAM system for 3D LiDARs. 1004–1011. https://doi.org/10.1109/IROS.2017.8202268

[25] Jakob Engel, Vladlen Koltun, and Daniel Cremers. 2018. Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 3 (2018), 611–625. https://doi.org/10.1109/TPAMI.2017.2658577

[26] T. Engelhardt, T. Konrad, Bjorn E. Schafer, and D. Abel. 2016. Flatness-Based Control for a Quadrotor Camera Helicopter using Model Predictive Control Trajectory Generation. *24th Mediterranean Conference on Control and Automation (MED)* (2016), 852–859.

[27] Eduardo Ferrera, Alfonso Alcántara, J. Capitán, Á. R. Castaño, P. Marrón, and A. Ollero. 2018. Decentralized 3D Collision Avoidance for Multiple UAVs in Outdoor Environments. *Sensors (Basel, Switzerland)* 18 (2018).

[28] Akshay Gadre, Revathy Narayanan, and Swarun Kumar. 2018. Maintaining UAV Stability Using Low-Power WANs. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (New Delhi, India) *(MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 738–740. https://doi.org/10.1145/3241539.3267747

[29] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. 2020. OpenVINS: A Research Platform for Visual-Inertial Estimation. In *Proc. of the IEEE International Conference on Robotics and Automation.* Paris, France. https://github.com/rpng/open_vins

[30] Shahram Ghandeharizadeh. 2021. Holodeck: Immersive 3D Displays Using Swarms of Flying Light Specks. In *ACM Multimedia Asia* (Gold Coast, Australia). https://doi.org/10.1145/3469877.3493698

[31] Shahram Ghandeharizadeh. 2022. Display of 3D Illuminations using Flying Light Specks. In *ACM Multimedia* (Lisboa, Portugal) *(MM '22)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3503161.3548250

[32] Shahram Ghandeharizadeh, Jason Yap, and Hieu Nguyen. 2014. Strong Consistency in Cache Augmented SQL Systems. In *Proceedings of the 15th International Middleware Conference* (Bordeaux, France) *(Middleware '14)*. Association for Computing Machinery, New York, NY, USA, 181–192. https://doi.org/10.1145/2663165.2663318

[33] C. Gray and D. Cheriton. 1989. Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency. *SIGOPS Oper. Syst. Rev.* 23, 5 (nov 1989), 202–210. https://doi.org/10.1145/74851.74870

[34] Bernhard Großiwindhager, Michael Stocker, Michael Rath, Carlo Alberto Boano, and Kay Römer. 2019. SnapLoc: An Ultra-Fast UWB-Based Indoor Localization System for an Unlimited Number of Tags. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 61–72.

[35] Yanying Gu, Anthony C. C. Lo, and Ignas G. Niemegeers. 2009. A Survey of Indoor Positioning Systems for Wireless Personal Networks. *IEEE Commun. Surv. Tutorials* 11, 1 (2009), 13–32. https://doi.org/10.1109/SURV.2009.090103

[36] Stephen. J. Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. 2009. ClearPath: Highly Parallel Collision Avoidance for Multi-Agent Simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New Orleans, Louisiana) *(SCA '09)*. Association for Computing Machinery, New York, NY, USA, 177–187. https://doi.org/10.1145/1599470.1599494

[37] Michael Hamer, Lino Widmer, and Raffaello Drandrea. 2018. Fast Generation of Collision-Free Trajectories for Robot Swarms Using GPU Acceleration. *IEEE Access* PP (12 2018), 1–1. https://doi.org/10.1109/ACCESS.2018.2889533

[38] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. 1993. Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, 9 (1993), 850–863. https://doi.org/10.1109/34.232073

[39] Hao Jiang, Tianlu Mao, Shuangyuan Wu, Mingliang Xu, and Zhaoqi Wang. 2014. A Local Evaluation Approach for Multi-Agent Navigation in Dynamic Scenarios. In *Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (Shenzhen, China) *(VRCAI '14)*. Association for Computing Machinery, New York, NY, USA, 89–93. https://doi.org/10.1145/2670473.2670493

[40] Roland Jung and Stephan Weiss. 2022. Scalable and Modular Ultra-Wideband Aided Inertial Navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2423–2430. https://doi.org/10.1109/IROS47612.2022.9981937

[41] Peng-Yuan Kao, Hsiu-Jui Chang, Kuan-Wei Tseng, Timothy Chen, He-Lin Luo, and Yi-Ping Hung. 2023. VIUNet: Deep Visual–Inertial–UWB Fusion for Indoor UAV Localization. *IEEE Access* 11 (2023), 61525–61534. https://doi.org/10.1109/ACCESS.2023.3279292

[42] Georg Klein and David Murray. 2007. Parallel Tracking and Mapping for Small AR Workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality*. 225–234. https://doi.org/10.1109/ISMAR.2007.4538852

[43] Sebastian Klose, Jian Wang, Michael Achtelik, Giorgio Panin, Florian Holzapfel, and Alois Knoll. 2010. Markerless, vision-assisted flight control of a quadrocopter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5712–5717. https://doi.org/10.1109/IROS.2010.5649019

[44] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. 2013. Towards a Swarm of Agile Micro Quadrotors. *Autonomous Robots* 35 (11 2013), 573–7527. https://doi.org/10.1007/s10514-013-9349-9

[45] Fabrizio Lazzari, Alice Buffi, Paolo Nepa, and Sandro Lazzari. 2017. Numerical Investigation of an UWB Localization Technique for Unmanned Aerial Vehicles in Outdoor Scenarios. *IEEE Sensors Journal* 17, 9 (2017), 2896–2903. https://doi.org/10.1109/JSEN.2017.2684817

[46] Anton Ledergerber, Michael Hamer, and Raffaello D'Andrea. 2015. A Robot Self-Localization System using One-Way Ultra-Wideband Communication. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 3131–3137. https://doi.org/10.1109/IROS.2015.7353810

[47] Leong Lee, Matthew Jones, Gregory S. Ridenour, Sam J. Bennett, Arisha C. Majors, Bianca L. Melito, and Michael J. Wilson. 2016. Comparison of Accuracy and Precision of GPS-Enabled Mobile Devices. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*. IEEE, 73–82. https://doi.org/10.1109/CIT.2016.94

[48] Jiarong Lin and Fu Zhang. 2020. Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 3126–3131. https://doi.org/10.1109/ICRA40945.2020.9197440

[49] Zhong Liu, Yuqing He, Liying Yang, and Jianda Han. 2017. Control Techniques of Tilt Rotor Unmanned Aerial Vehicle Systems: A Review. *Chinese Journal of Aeronautics* 30, 1 (2017), 135–148. https://doi.org/10.1016/j.cja.2016.11.001

[50] Nicola Macoir, Jan Bauwens, Bart Jooris, Ben Van Herbruggen, Jen Rossey, Jeroen Hoebeke, and Eli De Poorter. 2019. UWB Localization with Battery-Powered Wireless Backbone for Drone-Based Inventory Management. *Sensors* 19, 3 (2019). https://doi.org/10.3390/s19030467

[51] Luca Mainetti, Luigi Patrono, and Ilaria Sergi. 2014. A Survey on Indoor Positioning Systems. In *22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 111–120. https://doi.org/10.1109/SOFTCOM.2014.7039067

[52] Daniel Morgan, Soon-Jo Chung, and Fred Hadaegh. 2013. Decentralized Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming. In *Advances in the Astronautical Sciences*.

[53] Daniel Morgan, Soon-Jo Chung, and Fred Y. Hadaegh. 2014. Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming. *Journal of Guidance, Control, and Dynamics* 37, 6 (2014), 1725–1740. https://doi.org/10.2514/1.G000218

[54] Raúl Mur-Artal and Juan D. Tardós. 2017. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics* 33, 5 (2017), 1255–1262. https://doi.org/10.1109/TRO.2017.2705103

[55] Thulasi Mylvaganam, Mario Sassano, and Alessandro Astolfi. 2017. A Differential Game Approach to Multi-Agent Collision Avoidance. *IEEE Trans. Automat. Control* PP (04 2017), 4229–4235. https://doi.org/10.1109/TAC.2017.2680602

[56] Thien-Minh Nguyen, Muqing Cao, Shenghai Yuan, Yang Lyu, Thien Hoang Nguyen, and Lihua Xie. 2021. LIRO: Tightly Coupled Lidar-Inertia-Ranging Odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*. 14484–14490. https://doi.org/10.1109/ICRA48506.2021.9560954

[57] Michael Norman, Vince Kellen, Shava Smallen, Brian DeMeulle, Shawn Strande, Ed Lazowska, Naomi Alterman, Rob Fatland, Sarah Stone, Amanda Tan, Katherine Yelick, Eric Van Dusen, and James Mitchell. 2021. CloudBank: Managed Services to Simplify Cloud Access for Computer Science Research and Education. In *Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '21)*. Association for Computing Machinery, New York, NY, USA, Article 45, 4 pages. https://doi.org/10.1145/3437359.3465586

[58] Serban Georgica Obreja and Alexandru Vulpe. 2020. Evaluation of an Indoor Localization Solution Based on Bluetooth Low Energy Beacons. In *2020 13th International Conference on Communications (COMM)*. 227–231. https://doi.org/10.1109/COMM48946.2020.9141987

[59] Sohel J. Patel and Maciej J. Zawodniok. 2022. 3D Localization of RFID Antenna Tags Using Convolutional Neural Networks. *IEEE Transactions on Instrumentation and Measurement* 71 (2022), 1–11. https://doi.org/10.1109/TIM.2022.3146604

[60] Maxim Pavliv, Fabrizio Schiano, Christopher Reardon, Dario Floreano, and Giuseppe Loianno. 2021. Tracking and Relative Localization of Drone Swarms With a Vision-Based Headset. *IEEE Robotics and Automation Letters* 6, 2 (2021), 1455–1462. https://doi.org/10.1109/LRA.2021.3051565

[61] Trung Phan, Hamed Alimohammadzadeh, Heather Culbertson, and Shahram Ghandeharizadeh. 2023. An Evaluation of Three Distance Measurement Sensors for Flying Light Specks. In *International Conference on Intelligent Metaverse Technologies and Applications (iMETA)*.

[62] James Preiss, Wolfgang Hoenig, Nora Ayanian, and Gaurav Sukhatme. 2017. Downwash-Aware Trajectory Planning for Large Quadcopter Teams. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (04 2017), 8.

[63] James Preiss, Wolfgang Honig, Gaurav Sukhatme, and Nora Ayanian. 2017. Crazyswarm: A Large Nano-Quadcopter Swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*. 3299–3304. https://doi.org/10.1109/ICRA.2017.7989376

[64] Tong Qin, Peiliang Li, and Shaojie Shen. 2018. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics* 34, 4 (2018), 1004–1020. https://doi.org/10.1109/TRO.2018.2853729

[65] Guenther Retscher. 2020. Fundamental Concepts and Evolution of Wi-Fi User Localization: An Overview Based on Different Case Studies. *Sensors* 20, 18 (2020). https://doi.org/10.3390/s20185121

[66] Craig W. Reynolds. 1987. Flocks, Herds and Schools: A Distributed Behavioral Model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 25–34. https://doi.org/10.1145/37401.37406

[67] Craig W. Reynolds. 1987. Flocks, Herds and Schools: A Distributed Behavioral Model. *SIGGRAPH Comput. Graph.* 21, 4 (aug 1987), 25–34. https://doi.org/10.1145/37402.37406

[68] Robin Ritz, Mark W. Müller, Markus Hehn, and Raffaello D'Andrea. 2012. Cooperative Quadrocopter Ball Throwing and Catching. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4972–4978. https://doi.org/10.1109/IROS.2012.6385963

[69] Alina Rubina, Oleksandr Artemenko, Oleksandr Andryeyev, and Andreas Mitschele-Thiel. 2017. A Novel Hybrid Path Planning Algorithm for Localization in Wireless Networks. In *Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications* (Niagara Falls, New York, USA) *(DroNet '17)*. Association for Computing Machinery, New York, NY, USA, 13–16. https://doi.org/10.1145/3086439.3086441

[70] Nel Samama. 2008. *Global Positioning: Technologies and Performance.* https://doi.org/10.1002/9780470241912

[71] Martin Saska. 2015. MAV-swarms: Unmanned aerial vehicles stabilized along a given path using onboard relative localization. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. 894–903. https://doi.org/10.1109/ICUAS.2015.7152376

[72] Philip Shilane, Patrick Min, Michael M. Kazhdan, and Thomas A. Funkhouser. 2004. The Princeton Shape Benchmark. In *2004 International Conference on Shape Modeling and Applications (SMI 2004), 7-9 June 2004, Genova, Italy*. IEEE Computer Society, 167–178. https://doi.org/10.1109/SMI.2004.1314504

[73] Artur Shurin and Itzik Klein. 2022. QuadNet: A Hybrid Framework for Quadrotor Dead Reckoning. *Sensors* 22, 4 (2022). https://doi.org/10.3390/s22041426

[74] Bruno Silva, Zhibo Pang, Johan Åkerberg, Jonas Neander, and Gerhard Hancke. 2014. Experimental study of UWB-based high precision localization for industrial applications. In *2014 IEEE International Conference on Ultra-WideBand (ICUWB)*. 280–285. https://doi.org/10.1109/ICUWB.2014.6958993

[75] Haryong Song, Wonsub Choi, and Haedong Kim. 2016. Robust Vision-Based Relative-Localization Approach Using an RGB-Depth Camera and LiDAR Sensor Fusion. *IEEE Transactions on Industrial Electronics* 63, 6 (2016), 3725–3736. https://doi.org/10.1109/TIE.2016.2521346

[76] Martin Stephan, Javier Alonso-Mora, and Roland Siegwart. 2013. Reciprocal Collision Avoidance With Motion Continuity Constraints. *IEEE Transactions on Robotics* 29 (08 2013), 899–912. https://doi.org/10.1109/TRO.2013.2258733

[77] Yuan Sun. 2021. Autonomous Integrity Monitoring for Relative Navigation of Multiple Unmanned Aerial Vehicles. *Remote Sensing* 13, 8 (2021). https://doi.org/10.3390/rs13081483

[78] Camilla Tabasso, Venanzio Cichella, Syed Mehdi, Thiago Marinho, and Naira Hovakimyan. 2021. Time Coordination and Collision Avoidance Using Leader-Follower Strategies in Multi-Vehicle Missions. *Robotics* 10 (02 2021), 34. https://doi.org/10.3390/robotics10010034

[79] Xinyu Tong, Hao Li, Xiaohua Tian, and Xinbing Wang. 2021. Wi-Fi Localization Enabling Self-Calibration. *IEEE/ACM Transactions on Networking* 29, 2 (2021), 904–917. https://doi.org/10.1109/TNET.2021.3051998

[80] Roberto Tron, Justin Thomas, Giuseppe Loianno, Kostas Daniilidis, and Vijay Kumar. 2016. A Distributed Optimization Framework for Localization and Formation Control: Applications to Vision-Based Measurements. *IEEE Control Systems Magazine* 36, 4 (2016), 22–44. https://doi.org/10.1109/MCS.2016.2558401

[81] Jur van den Berg, Stephen Guy, Ming Lin, and Dinesh Manocha. 2011. *Reciprocal n-Body Collision Avoidance*. Vol. 70. 3–19. https://doi.org/10.1007/978-3-642-19457-3_1

[82] Jur van den Berg, Ming Lin, and Dinesh Manocha. 2008. Reciprocal Velocity Obstacles for Real-Time Multi-agent Navigation. *ICRA*, 1928–1935. https://doi.org/10.1109/ROBOT.2008.4543489

[83] Jur van den Berg, Jamie Snape, Stephen Guy, and Dinesh Manocha. 2011. Reciprocal Collision Avoidance with Acceleration-Velocity Obstacles. 3475–3482. https://doi.org/10.1109/ICRA.2011.5980408

[84] Jur van den Berg, David Wilkie, Stephen Guy, Marc Niethammer, and Dinesh Manocha. 2012. LQG-obstacles: Feedback Control with Collision Avoidance for Mobile Robots with Motion and Sensing Uncertainty. *Proceedings - IEEE International Conference on Robotics and Automation* (05 2012), 346–353. https://doi.org/10.1109/ICRA.2012.6224648

[85] Quoc Van Tran and Hyo-Sung Ahn. 2020. Distributed Formation Control of Mobile Agents via Global Orientation Estimation. *IEEE Transactions on Control of Network Systems* 7, 4 (2020), 1654–1664. https://doi.org/10.1109/TCNS.2020.2993253

[86] Bowen Wang, Yunlong Wang, Xinyou Qiu, and Yuan Shen. 2021. BLE Localization With Polarization Sensitive Array. *IEEE Wireless Communications Letters* 10, 5 (2021), 1014–1017. https://doi.org/10.1109/LWC.2021.3055558

[87] Han Wang, Chen Wang, Chun-Lin Chen, and Lihua Xie. 2021. F-LOAM: Fast LiDAR Odometry and Mapping. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Prague, Czech Republic). IEEE Press, 4390–4396. https://doi.org/10.1109/IROS51168.2021.9636655

[88] Xinjie Wang, Jiaping Ren, Xiaogang Jin, and Dinesh Manocha. 2015. BSwarm: Biologically-Plausible Dynamics Model of Insect Swarms. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '15)*. Association for Computing Machinery, New York, NY, USA, 111–118. https://doi.org/10.1145/2786784.2786790

[89] Yixin Wang, Qiang Ye, Jie Cheng, and Lei Wang. 2015. RSSI-Based Bluetooth Indoor Localization. In *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. 165–171. https://doi.org/10.1109/MSN.2015.14

[90] Aaron Weinstein, Adam Cho, Giuseppe Loianno, and Vijay Kumar. 2018. Visual Inertial Odometry Swarm: An Autonomous Swarm of Vision-Based Quadrotors. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1801–1807. https://doi.org/10.1109/LRA.2018.2800119

[91] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. 2002. An Integrated Experimental Environment for Distributed Systems and Networks. *SIGOPS Oper. Syst. Rev.* 36, SI (Dec. 2002), 255–270. https://doi.org/10.1145/844128.844152

[92] Kamin Whitehouse, Chris Karlof, and David Culler. 2007. A Practical Evaluation of Radio Signal Strength for Ranging-Based Localization. *SIGMOBILE Mob. Comput. Commun. Rev.* 11, 1 (jan 2007), 41–52. https://doi.org/10.1145/1234822.1234829

[93] Jinhui Wu, Dequan Zhang, Jie Liu, and Xu Han. 2020. A Moment Approach to Positioning Accuracy Reliability Analysis for Industrial Robots. *IEEE Transactions on Reliability* 69, 2 (2020), 699–714. https://doi.org/10.1109/TR.2019.2919540

[94] Fei Xiao, Shengkai Zhang, Sheyang Tang, Shaojie Shen, Huixin Dong, and Yi Zhong. 2023. WiSion: Bolstering MAV 3D Indoor State Estimation by Embracing Multipath of WiFi. *IEEE Transactions on Vehicular Technology* 72, 1 (2023), 253–266. https://doi.org/10.1109/TVT.2022.3205710

[95] Yaxiong Xie, Jie Xiong, Mo Li, and Kyle Jamieson. 2018. mD-Track: Leveraging Multi-Dimensionality for Passive Indoor Wi-Fi Tracking. *The 25th Annual International Conference on Mobile Computing and Networking* (2018).

[96] Jie Xiong and Kyle Jamieson. 2013. ArrayTrack: A Fine-Grained Indoor Location System. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation* (Lombard, IL) *(nsdi'13)*. USENIX Association, USA, 71–84.

[97] Hao Xu, Luqi Wang, Yichen Zhang, Kejie Qiu, and Shaojie Shen. 2020. Decentralized Visual-Inertial-UWB Fusion for Relative State Estimation of Aerial Swarm. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 8776–8782. https://doi.org/10.1109/ICRA40945.2020.9196944

[98] Hao Xu, Yichen Zhang, Boyu Zhou, Luqi Wang, Xinjie Yao, Guotao Meng, and Shaojie Shen. 2022. Omni-Swarm: A Decentralized Omnidirectional Visual-Inertial-UWB State Estimation System for Aerial Swarms. *IEEE Trans. Robotics* 38, 6 (2022), 3374–3394. https://doi.org/10.1109/TRO.2022.3182503

[99] Yang Xu, Shupeng Lai, Jiaxin Li, Delin Luo, and Yancheng You. 2019. Concurrent Optimal Trajectory Planning for Indoor Quadrotor Formation Switching. *Journal of Intelligent & Robotic Systems* 94 (05 2019). https://doi.org/10.1007/s10846-018-0813-9

[100] Bo Yang, Jun Li, and Hong Zhang. 2021. Resilient Indoor Localization System Based on UWB and Visual–Inertial Sensors for Complex Environments. *IEEE Transactions on Instrumentation and Measurement* 70 (2021), 1–14. https://doi.org/10.1109/TIM.2021.3101322

[101] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. 2019. A Survey of Indoor Localization Systems and Technologies. *IEEE Communications Surveys and Tutorials* 21, 3 (2019), 2568–2599. https://doi.org/10.1109/COMST.2019.2911558

[102] Zebra. 2018. Zebra UWB Technology, December 1, 2018, https://bit.ly/3Fji5Bs.

[103] Jitong Zhang, Mingrong Ren, Pu Wang, Juan Meng, and Yuman Mu. 2020. Indoor Localization Based on VIO System and Three-Dimensional Map Matching. *Sensors* 20, 10 (2020). https://doi.org/10.3390/s20102790

[104] Ji Zhang and Sanjiv Singh. 2014. LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, Dieter Fox, Lydia E. Kavraki, and Hanna Kurniawati (Eds.). https://doi.org/10.15607/RSS.2014.X.007

[105] Jian Zhang, Xiangyu Wang, Zhitao Yu, Yibo Lyu, Shiwen Mao, Senthilkumar CG Periaswamy, Justin Patton, and Xuyu Wang. 2019. Robust RFID Based 6-DoF Localization for Unmanned Aerial Vehicles. *IEEE Access* 7 (2019), 77348–77361. https://doi.org/10.1109/ACCESS.2019.2922211

[106] Shiyu Zhao, Zhenhong Li, and Zhengtao Ding. 2019. Bearing-Only Formation Tracking Control of Multiagent Systems. *IEEE Trans. Automat. Control* 64, 11 (2019), 4541–4554. https://doi.org/10.1109/TAC.2019.2903290

[107] Cao Zhiqiang, Ran Liu, Chau Yuen, Achala Athukorala, Benny Ng, Muraleetharan Mathanraj, and U-Xuan Tan. 2021. Relative Localization of Mobile Robots with Multiple Ultra-WideBand Ranging Measurements. 5857–5863. https://doi.org/10.1109/IROS51168.2021.9636017